DIJITAL LOGIC DESIGN CSE3015 PROJECT REPORT

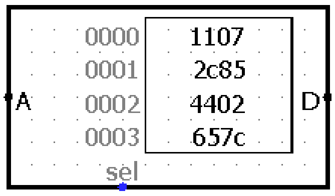
Ezgi Doğruer	İsra Nur Alperen	Elif Gökpınar

We implemented a processor that includes AND, ADD, LD, ST, ANDI, ADDI, CMP, JUMP, JE, JA, JB, JBE, JAE instructions.

Properties

- Processor will have 16 bits address width and 16 bits data width.
- 8 registers are used in processor.(R0,R1,R2,…..,R7)
- Data Memory has 10 bits address width and 16 bits data width.
- There are 5 components : Instruction memory, Data Memory ,Register File, ALU, Control Unit.

Instruction Memory



Instruction memory is only read only memory.ROM is used for reading instructions. It can not be changed.

PC

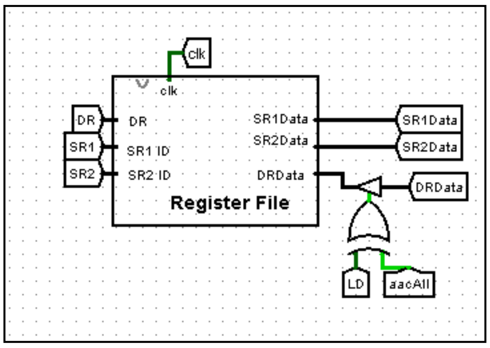
FullData : It is the full data which comes from Instruction memory.

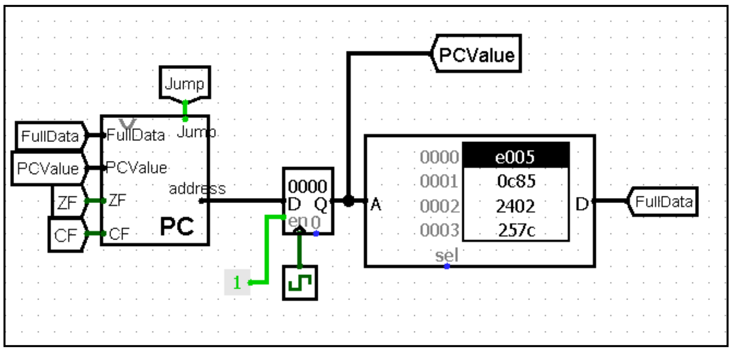
PCValue : It holds the next instruction address.

ZF : Zero Flag.

CF : Carry Flag.

If there is a jump signal, ZF and CF are controlled by their type of jmp.(JE,JA,JAE,JB,JBE).If their special conditions are true, instructions are implemented. PC changes with given value. Value is assigned to the register every rising edge.





If there is a jump signal, the PC uses jump instructions if their conditions are true. If there is no jump signal, the PC will work normally. It increments the next address by 1.

Register File

DRData : The new value comes the register file.

DR : DR is given the register file. Then the new value is assigned to the this register.

SR1 : It holds the SRC 1 ID’s.

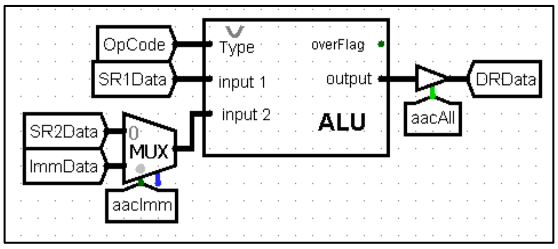
SR2 : It holds the SRC 2 ID’s if it exists.If there is immediate value,ReadR2 is not read.

SR1Data : It holds the SRC 1 address.

SR2Data : It holds the SRC 1 address if it exists.

ALU

4 arithmetic instructions are executed in ALU: AND, ANDI, ADD, ADDI.

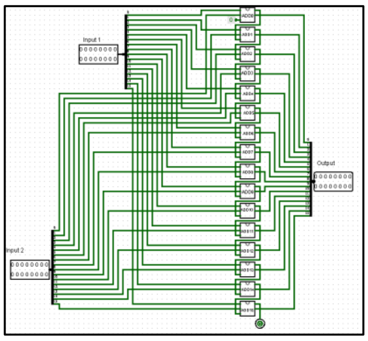


For ADD and ADDI instructions, created ADDER and Full Adder are used. Also we generated a AndAddControl to control immediate value or SR2. ALU takes opcode as a Type input to decide is this ADD or AND instruction. Also takes SR1Data as input1 and takes SR2 if instruction is not ADDI or ANDI, otherwise takes ImmData that we take from AndAddContorl in Control Unit as input 2. Output is 16 bit output to store DR. aacAll is control if this ADD or ADDI or AND or ANDI.

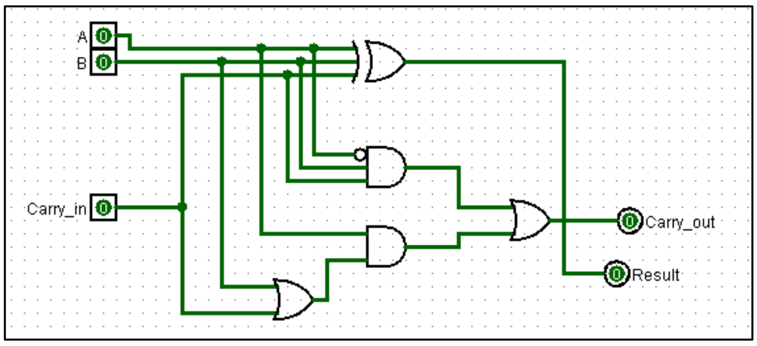
ADDER

We generated Adder to add two 16bit variables in ALU. In addition we generated Full Adder to generate Adder.

Adder:



Full Adder:



DATA MEMORY

Data is stored in RAM and can be retrieved and changed.

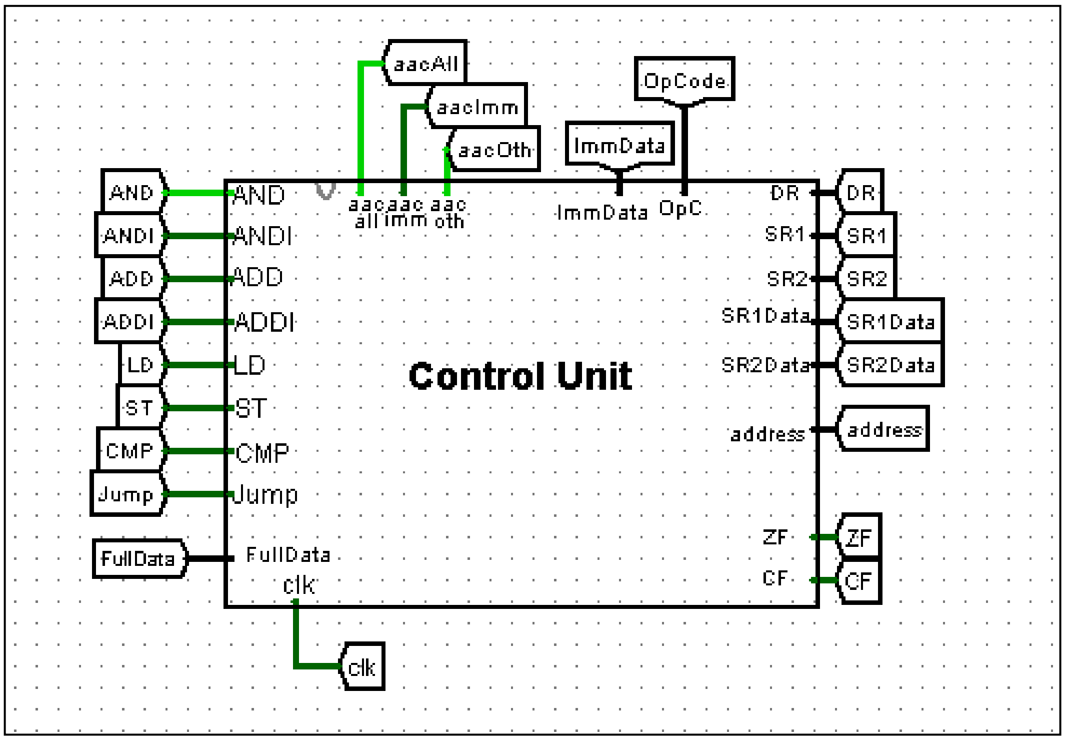
It is controlled by the ST signal, LD signal, and the clock signal. These signals are sent from the Control Unit.

When the ST instruction is used, the control unit allows data to be stored via the ST signal. *SR1Data* goes to the given address. RAM stores the SR1 data.

If LD instruction is used, Control unit permits the load data via LD signal. RAM loads the data at a given address. This data contains the new content of the given register in the instruction.

CONTROL UNIT

A control unit coordinates how data moves around a CPU. It manages the components via signals. *CMP,LD,ST,JMP* instructions, zero flag and carry flag control are coordinated by CU.

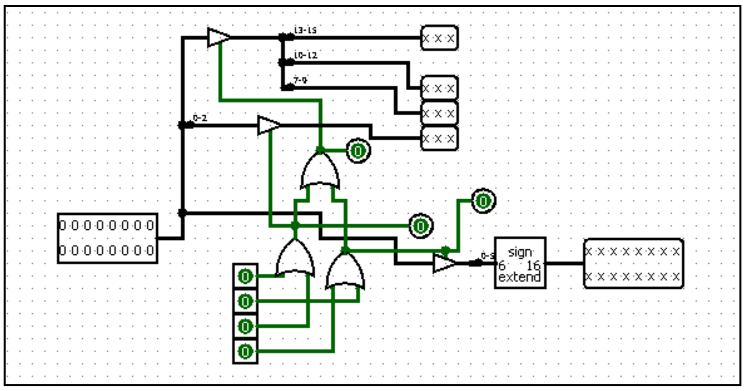


We generated tunnel for the instructions and assigned signals to these tunnels. Likewise, we have generated values like address *sr1 data, sr2 data* and *immdata* that we will use for instructions.

Firstly, we took the opcodes of our data and assigned them to a decode and sent signals from here to the instructions that need to be run. In the control unit, we have created circuits that perform AndAdd control, LD, ST and CMP instruction operations. We output the results of the instructions inside the control unit circuit as an output.

AAC

AAC is AndAddControl. We created this component to control if this use SR2 or Immediate Value. If this is ADDI or ANDI, it takes immediate value and extend it to 16 bits. If not sends SR2 value.



CMP

In CMP part we generated 3 components as 1 bit comparator, 16 bit comparator and comparator.

1 Bit Comparator

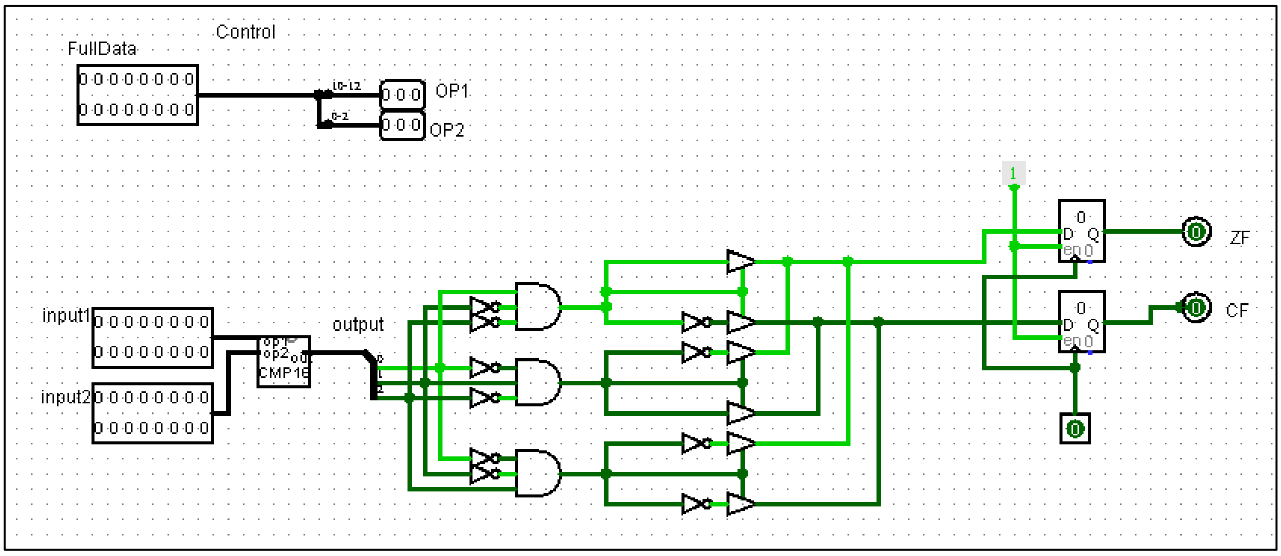
We took li (less than input), gi (greater than input), A and B. *Li* and *gi* to takes information from previous bit comparator. This component sends *lt_output*, *gt_output* and *eq_output* as output.

16 Bit Comparator

In this component we compared 16 bits value using 1 bit comparator. Finally, we took 3 bits output if input1 and input 2 are equal, input1 less than or input1 greater than.

Comparator

In this component we splitted *Full Data* to find *SR1* and *SR2*. After that we send input 1 and input 2 to 16 bits comparator and assign ZF and CF according to output of 16 bits comparator.

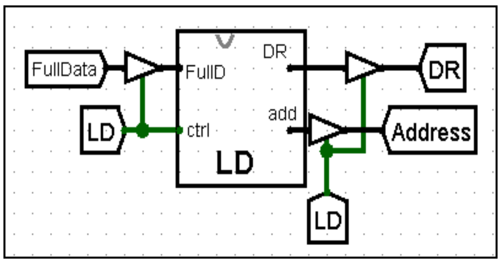


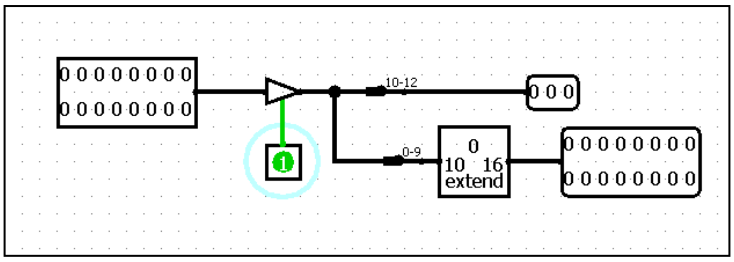
LD

FullD : It is the full data which comes from Instruction memory.

LD : It is signal that is produced in Control unit.

DR : It is the destination register id.For example: R0 -> 000 , R1 -> 001.





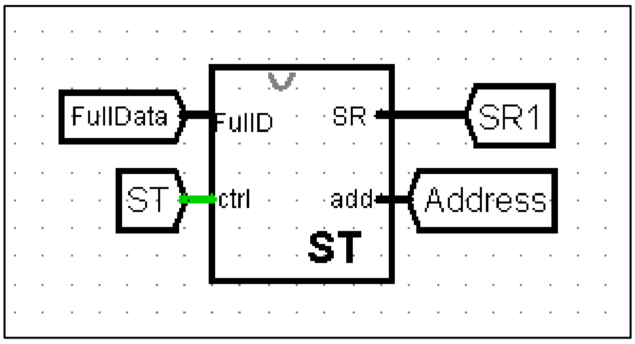
Address: Last 10 bits of full data shows the memory address, it extends the 16 bit. This memory address goes to the Data memory and fetch the information of this address. *DRData* takes the this address. This address is record in the register file to DR register.

In order to this, Control unit must send the LD signal.

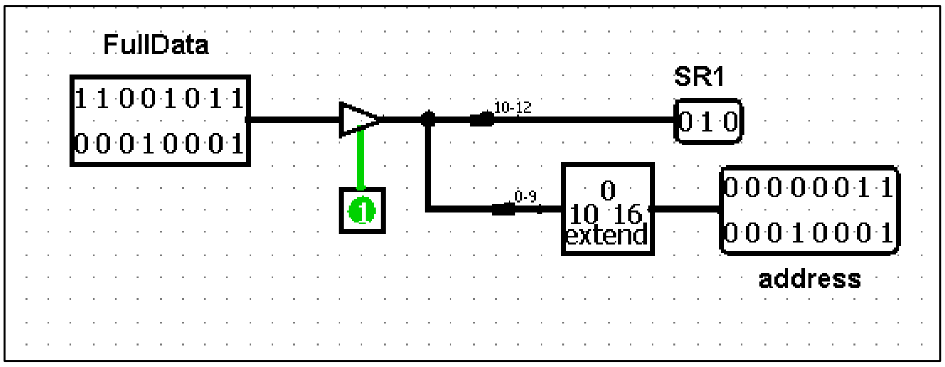
ST

We do the ST instruction operations in the control unit. First, we assigned a signal just like we did in other instructions. We send the data to the ST circuit from the part where we receive and distribute it from the decoder. We took our data and str signal as inputs to the circuit we created for the ST operation. In the circuit, we send the source register part (between 10 and 12 bits) of the data we received to the SR1 tunnel. Also, we extended the rest of the data to 16 bits and send it to the address tunnel. The address and source register output from the ST circuit are used in data memory. Therefore, actual operations are performed in data memory. In data memory, the value in the SR register from the register file is written to the address we send and the ST instruction completes.

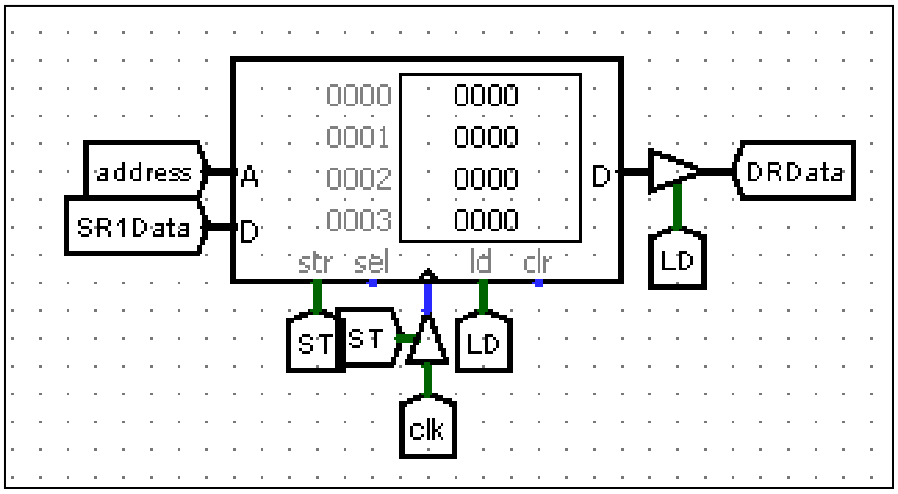
ST



Inside of ST



Data Memory


0000 0000
0001 0000
0002 0000
0003 0000
address
A
SR1Data
D
D
DRData
LD
str
sel
ld
clr
ST
ST
LD
clk